Z-100 LifeLine
Since 1989

March 2022

#WEB

This article was first published in issue #100, August 2005
~~~~~~

DSKCOPY4.COM Documentation

by Steven Vagts
Editor, "Z-100 LifeLine"

DSKCOPY4.COM

Introduction:

Way back with the desire to upgrade to DOS v4 and the introduction of the newer ROMs to make this possible, it was realized that the standard DOS v4 DISKCOPY was going to need upgrading. The new DISKCOPY would need to deal with all the existing Heath/Zenith floppy disk formats, of which the standard Microsoft-DOS DISKCOPY wasn't aware, and the newer capabilities being added to the Z-100.

The DISKCOPY distributed with Z-DOS v4 was an interim version that relied heavily on using the then current version of FORMAT.COM for its low level formatting functions. However, as FORMAT changed over the years, with new and different switches and new disk formats, DISKCOPY no longer knew how to properly execute FORMAT properly. A much more robust version of DISKCOPY was needed.

About that same time, it was decided that with the decreasing reliability of floppy disks, the new program needed to make every possible effort to recover data from problem diskettes and include a method of archiving and re-creating bootable floppy disks. It was conceived that an image of a boot disk could be saved as a disk file on a hard drive, and any time that a new bootable floppy disk was needed, the program could reverse the process and generate a fresh copy of the needed disk.

John Beyers began working diligently on the new upgraded requirements and made a suitable replacement that was essentially completed as DSKCOPY4 back in 2002! The program could:

* Read nearly all 8", 5.25", and 3.5" soft-sectored disk formats, including all versions of CP/M and MS-DOS
* Support both FM & MFM density formats, although they cannot be mixed on any given side of a single track
* Create identical copies of 8", 5.25", and 3.5" disks
* Save the entire disk structure as a disk image file
* Re-create the floppy disk from the disk image file
* Decode the disk structure, permitting a user to:
  - Salvage data from some damaged disks
  - Recognize and reconstruct missing sectors
  - Correct bad sector marks
* Create a 5-1/4" or 3-1/2" High Density disk from an 8" disk

Note: This last feature was possible because a 5-1/4" disk shares the same parameters as an 8" drive (and they both spin at 360 RPM), but with 80 tracks, instead of 77. DSKCOPY4 can make an image of a 77 track 8" floppy disk and place the data on an 80 track 5-1/4" drive by adding three additional blank tracks. You can also create the 77 track simulation on a 3-1/2" disk.

However, to fully test the additional features of DSKCOPY4, John wanted to try the program on as many disk formats as was available to him, hence the reason for ZFMT207 reviewed extensively in the "*Z-100 LifeLine*" Issue #99. The new format program could create all kinds of weird bootable formats with which to test DSKCOPY4.

The new capabilities of DSKCOPY4 are therefore the subject of this article.
~~~~~~

Limitations:

* DSKCOPY4 requires MTR-ROM v4.3 and Z-DOS v4.06 or later.

* DSKCOPY4 cannot be used on hard-sectored diskettes; the Z-100 cannot use hard-sectored drives.

* DSKCOPY4 requires that destination disks be about 5% larger than the source disks when making disk image files.

DSKCOPY4's Help Screen

As with most of the Z-DOS v4 Commands, the Help Screen can be displayed by simply typing the Command followed by a space and question mark; or a space, forward slash mark, and question mark; followed by pressing the {RETURN} key. Therefore, if you use either command;

DSKCOPY4 ?{RETURN}
DSKCOPY4 /?{RETURN}

the DSKCOPY4 Help Message of Figure 1 is displayed.

~~~~~~

```
DSKCOPY4 Version 4.06 - Command line format:
     DSKCOPY4 [?] [/x][source][ destination][/x]
 where /x is any combination of the following switches:
  /A   Inspect All SOURCE tracks, not just Track 0
  /C   Automatically accept data on CRC read errors
  /D   Convert DELETED data address marks
  /F   Use FM mode on unknown SOURCE tracks
  /I   Infinite retries on CRC read errors
  /N   Automatically accept data on Record NOT Found errors
  /Q   Query for SOURCE physical characteristics
  /Rn  Retry n times on READ errors
  /S   Automatically SKIP extra sectors found with /X switch
  /U   Automatically accept Unformatted tracks
  /V   Do NOT Verify on DESTINATION disk
  /X   Perform eXtended SOURCE track inspection
  /Y   Do NOT Format on DESTINATION disk
  /1   Erase Side 1 on Single Sided disk
Source/destination can be either a drive letter or a filespec. If both are
drive letters, no disk file comes into play. If no/partial/incorrect command
line is given, you will be prompted for the incomplete/incorrect information.
DSKCOPY4 requires DOS/BIOS version 4.06 or later.
"Z-100 LifeLine" issue #100 has a complete review of DSKCOPY4.
```

Figure 1.
DSKCOPY4's Help Screen

~~~~~~

The Help Screen can be printed out by using IO redirection (PRN or NUL); for example:

DSKCOPY4 /? >PRN{RETURN}

The DSKCOPY4 program is invoked by:

Usage: [a:\][pathname\]**DSKCOPY4** [?]
       [/x][source][destination][/x]

Where anything in [brackets] is optional and:

a:\path - Before command is the drive\ pathname where the transient command file is located.
[source] - Is the drive identifying the source drive.
[destn] - Is the drive identifying the destination drive.

Note: The [destination] can be an entire d:\pathname\filename where a file image of the entire source disk can be stored.

DSKCOPY4 uses no compression scheme, so ensure that the destination drive containing this new image file has enough free disk space for about 105% of the source disk's formatted capacity. See the "Disk Image File Format" in the **Reference Information** at the end of this article for more information.

You may want to ZIP the resulting disk file so it will take up less space. Just for comparison, a typical 360K disk filled with text files uses approximately 362K unzipped and only 87K when zipped with a program like WINZIP or PKZIP. A disk filled with program files will zip to about 300K.

/x - Is any combination of the following switches:

Note: Many of the switches have been created to deal with the expected problems associated with bad sectors on old disks. Basically, we would like to give our best shot at getting as much data as we can.

/A - Inspect All SOURCE tracks, not just Track 0

Selecting this switch will dramatically increase the amount of time required to read a SOURCE disk. Without this switch, it is assumed all tracks are formatted exactly the same as Track 0 Side 1 for double sided media; or Track 1 Side 0 for single sided media (base format).

Note: We ignore similarity for Track 0 Side 0 because some CP/M disks have FM on Track 0 Side 0 and MFM on the rest of the disk.

We also have a dummy, 128-byte sector 16 on High Density 5 1/4" disks with 512 byte sectors on each track. While 8" disks have 16 sectors per track, 5-1/4" high density disks only have 15 sectors. When used as boot disks, these 15 sector disks generated an error. So FORMAT.COM distributed with earlier Z-DOS v4 versions created 5-1/4" high density boot disks that added an extra 128 byte 16th sector to eliminate the error. The new v4.3 MTR-ROM has eliminated this requirement.

Most DOS and CP/M disks should copy fine without the /A switch.

"Copy Protected" and proprietary data disks will probably not copy correctly without this switch and possibly the /X switch. Some problems are detected and errors occur without this switch, but unfortunately some differences go undetected when a track is different from the base format. The undetected errors will occur because:

- Extra Sectors on a track are missed and not copied
- Smaller Sector Sizes are padded
- Larger Sector Sizes give a Lost Data error.

/C - Accept data on CRC READ errors

This might be useful when a sector goes bad and you wish to attempt salvaging as much information as possible. If it happens to be only text, you may be able to fix either the disk file or the newly created disk with an editor.

/D - Convert DELETED data address marks

The Data Address Mark (DAM) is the last byte of the inter-sector gap between sector data fields and indicates the start of the data field. The 1797 chip on the Z-207 card only recognizes two possible DAM bytes and there is only a one bit difference between a regular DAM and a deleted DAM byte. If the bit is changed, whether on purpose by a user application, or by being damaged by age, or physical abuse, the mark may be treated as bad and deleted.

Additionally, if a DAM has been so damaged that it is neither of the recognized bytes, that sector's data is lost because the start of the data field cannot be determined.

It is not certain if any usable disks actually have deleted data address marks. But since the Z-207 card is capable of recognizing deleted DAMs, we took advantage of this capability. So, if the /D switch is used, any deleted DAM found will be modified to the regular DAM, making that sector readable.

While older BIOS's flagged a deleted DAM as an error and would skip that sector of data, the new v4 BIOS does not. The new BIOS notifies us that it found a deleted DAM and will continue processing the data.

/F - Use FM mode on unknown tracks

The default is to attempt the READ TRACK command using MFM mode when the READ ADDRESS command fails to find a valid sector.

/I - Infinite retries on READ errors

If you happen to have only one copy of some critical data or software and some sector(s) on the disk go bad, using this switch may enable recovery. Some bad sectors have been recovered after more than 100 retries.

/N - Automatically accept data on Record NOT Found errors

Normally, if the program cannot find a certain sector, it will ask you if you want to accept the data when a record is not found. This switch will force the acceptance of data automatically for any records not found rather than asking for each instance.

/Q - Query for SOURCE physical characteristics

This switch will prompt you for the number of tracks, number of sides, High(1.2Meg) or Low(360/720K) Density media, and Double Stepping. It could be useful on non-DOS disks that have errors on the first track.

/Rn - Retry n times on READ errors

This switch allows you to select the number of retries on detection of a bad sector. Valid values are 1 to 254. The default is no retries.

/S - Automatically SKIP extra sectors found with /X switch

This switch only comes into play if the /X switch has also been selected. It will suppress the question for Adding or Skipping an extra sector that was found by the READ TRACK command, and assumes you want to skip it.

Since the sector was not found by the READ ADDRESS command, it is probably not really a sector, but just data - possibly because its ID header is so bad that the READ ADDRESS Command could not find it, but READ TRACK did. In that case, we can choose Add. This will not allow READ SECTOR access, which requires needing an address, but we could accept the data from READ TRACK.

/U - Automatically accept Unformatted Tracks

This will suppress the question for accepting unformatted tracks and assume yes.

/V - Do NOT Verify on DESTINATION disk

The Default was chosen to have verify ON because the Write Sector command really has no way to tell if the data was written correctly. You need to do a verify to ensure the CRC is correct. If it fails, DSKCOPY4 aborts with an error.

/Y - Do NOT Format on DESTINATION disk

This switch was created so it might be possible to create a disk that is useable when the source disk is so messed up that not all the sectors on a track can be located. When this happens those lost sectors will not be created on the destination disk if you reformat.

When you do not format the destination, the sectors you DO get from the source will be put on the destination disk and the lost ones will be empty. Of course using this switch requires that the format of the destination matches the source (number of tracks, number of sides, FM /MFM, and number of sectors per track) exactly.

/X - Perform eXtended track inspection

This will force both READ ADDRESS and READ TRACK to be performed on track data. The program will attempt to see if the sector size defined by the ID header is actually on the track. If it looks like there has been an attempt to create a partial sector, the program will TRY to re-create it.

Selecting this switch will increase the amount of time required for the copy process if the /A switch is selected.

/1 - Erase Side 1 on Single Sided disk

This switch will erase side 1 on single sided disks so no previously placed valid information is left on the unused side. Only use this switch when you are creating single sided media using a double sided drive and media.

Operational Examples:

Now let's get into the practical use of the program.

Standard Copy:

Let's begin with a normal disk copy, say from drive A: to drive B:. As with the older versions of DISKCOPY, the source and destination must be of the same type and size. In this case, they are normal 360K drives. Enter the command:

DSKCOPY4 A: B:{RETURN}

The computer will respond with:

Please insert source/destination disk(s) and hit any key to continue....

40 Tracks, Double Sided, 250K Transfer, 300rpm
Reading Source
Track XX Side X
Writing Destination
Track XX Side X (action)
Copy completed successfully.

Note: The (action) above will alternate between displaying 'Formatting', 'Writing", and 'Verifying'.

Note: If you neglect to add the drive letters after the DSKCOPY4 command, you will be prompted individually for the source drive and then the destination drive.

If you made a mistake and the drives were not identical, you can expect to see an error. For example, drive B: on my computer was a high density 3.5" drive, and I got the error:

Destination device has at least twice as many tracks as source media. If you are attempting to create 40 track media in an 80 track device by double stepping, there is no guarantee of success if you are planning on reading it in a 40 track device.
You may also use only the first 40 tracks without double stepping.
Use ^C to Quit. Double Step media?(Y or N)>

Note: ^C is {Control}-{C}. If you press {Control}-{C} to quit, the computer will respond with:

DSKCOPY4 aborted by user, copy incomplete!

If you made a mistake and tried to copy to the same drive, you can expect to see the error:

Source and Destination can not be the same drive.
Use DRIVECFG to setup imaginary drive and try again.
Please input destination drive or filename:

So as you can see, John Beyers tried to make the program as helpful and user responsive as possible.

Using an Imaginary Drive:

Since we're on the subject, let's pursue use of an Imaginary Drive further. If you are like me, you will probably have several drives attached to the computer, but only one of each type. To make a copy of a 360K floppy drive to another 360K floppy, would require using an imaginary drive.

If you haven't already assigned an Imaginary Drive, we need to do that first, by using DRIVECFG to assign one. So, from within the root directory, enter the command:

DRIVECFG{RETURN}

The DRIVECFG screen will list all the present drives, up to 26 drive letters, in the computer. Highlight the **"Next Available Drive to Assign"** and press {RETURN}. A window will open that lists the various drive types.

Select "**(F0) Imaginary Drive**" and press {RETURN} again.

The computer will respond with:

Select Physical Drive Letter to map this Imaginary Device (ESC to cancel)...

Enter the drive letter of the Physical Drive that will be used. I assigned Imaginary Drive J: to Physical Drive A:.

You are returned to the drive listing and will see that an Imaginary Drive is now listed on the screen. Press {ESCape} and save the changes. Press {ESCape} again to quit. Finally, reboot the computer for the changes to take effect.

Note: John Beyers has also written a small utility that permits easily changing the imaginary drive's physical drive assignment. See the section, **"Using SETIMAG"**, below for further information.

~~~~~~

CAUTION: Use imaginary drives very carefully. As DSKCOPY4 has no real knowledge of your intended disk/drive combination, it is your responsibility to ensure the right disk is in the drive to perform your intended operation. While the program does its best to stay on top of what is where, the potential is there to end up copying the wrong information to the wrong disk, especially between commands.

If a **previous program** finishes with a certain disk as Imaginary, before the **next command** is issued, either modify the command's drive combination to reflect the actual disk in the drive, or force a disk change using the directory command.

~~~~~~

Now that we have assigned a drive letter to an Imaginary Drive, in order to make a copy of a 360K floppy using a single drive, we can use the command:

DSKCOPY4 A: J:{RETURN}

Where J: is our newly assigned Imaginary Drive, using Physical Drive A:.

The computer will respond with:

Please insert source/destination disk(s) and hit any key to continue....

On the right center of the screen, a graphic, reverse video ICON (Figure 2) in the form of a disk drive will appear with drive letters to show when to change disks in the drive. When the **[J->A]** ICON appears, it means to "Place disk J: in drive A: and press {RETURN}".

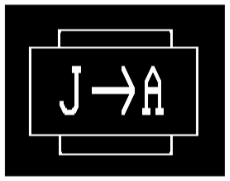

Figure 2.
Disk Change ICON

40 Tracks, Double Sided, 250K Transfer, 300rpm

The **[A->A]** ICON now appears. Place disk A: back into drive A: and press {RETURN}.

Reading Source
Track XX Side X

When the **[J->A]** ICON appears, place disk J: in drive A: again and press {RETURN}.

Writing Destination
Track XX Side X (action)
Copy completed successfully.

Note: The (action) above will alternate between displaying 'Formatting', 'Writing' and 'Verifying'. You will notice that 'Writing' is often not displayed. This is because the data stream to be written is screened for certain data bytes that are used as control characters. If there are no special bytes, the data is written during the formatting, and thus saving considerable time. If there are special bytes, the data has to be written separately.

The program finishes with disk J: in the drive. If you want to switch back to disk A:, the easiest action would be to enter the DIR A: Command. The **[A->A]** ICON will reappear. Simply place disk A: in drive A: and press {RETURN}.

Creating a Disk Image:

We can create a disk image and save it as a file to any drive with a larger capacity - about 5% greater than the diskette having an image made. The destination could be a hard drive, a RAM Disk (such as a 1Mb Z-205 card), another floppy drive, or even the same drive, if:

* the source disk was an odd FM format and the destination was an MFM format, or
* the source disk was single sided and the destination was double sided.

Note: Remember, you cannot use the same drive letter; you must assign an imaginary drive letter to one of the disks.

Let's begin by creating a disk file image of drive A: on the hard drive, drive E:, using a command similar to:

```
    DSKCOPY4 A: E:\Image\(filename){RETURN}
```

In this example, we save (filename) in a subdirectory called \Image.

The computer will respond with:

Please insert source/destination disk(s) and hit any key to continue....

40 Tracks, Double Sided, 250K Transfer, 300rpm
Please enter up to 79 character Expanded Disk File Label:

As we could end up with images of dozens of disks on a hard drive, we wanted a better way to describe a disk image than a very limiting 8 character filename. So, John included the ability to add a more descriptive label to the beginning of the disk file image. It can be displayed by a couple of methods. See the section, **"Using Expanded Disk File Labels"**, below.

A suggested description might include information about the source disk, such as:

5.25" CP/M-85 Boot Disk.

The computer continues with:

Reading Source
Track XX Side X
Writing Destination
E:\Image\(filename)
Copy completed successfully.

Note: If you neglect to add the drive letters after the DSKCOPY4 command, you will be individually prompted for the source drive and then the destination drive or file name.

As explained earlier, the destination could just as easily be an imaginary drive, as long as the destination has a greater capacity.

For example, if we had a dual density, physical drive C:, such as a 5.25" 360K/1.2Mb drive, or a 3.5" 720K/1.4Mb drive, we could assign an Imaginary Drive Letter J: to the dual density drive (using DRIVECFG or SETIMAG) and then use the command:

```
    DSKCOPY4 C: J:(filename){RETURN}
```

A disk file image is then made of the lower density disk in the C: drive and is saved on a high density disk in the same drive with imaginary drive letter J:.

The 713K disk becomes a 730K file.

If you happen to forget to add the (filename), you will see the error:

Destination drive/media INCOMPATIBLE with SOURCE media!

Re-creating a Disk from a Disk Image:

We can re-create the disk from a disk image file. The destination drive must be the same type drive and media from which the image was made, with one exception. We could simulate the disk on any larger size media, a topic we'll discuss in the next section!

So to re-create a copy of the disk from a disk image created in the previous example, use a command similar to:

```
    DSKCOPY4 E:\Image\(filename) A:{RETURN}
```

The computer will respond with:

Please insert source/destination disk(s) and hit any key to continue....

40 Tracks, Double Sided, 250K Transfer, 300rpm
Reading Source
E:\Image\(filename)
Writing Destination
Track XX Side X (action)
Copy completed successfully.

Note: The (action) above will alternate between displaying 'Formatting', 'Writing", and 'Verifying'.

Note: If you neglect to add the drive letters after the DSKCOPY4 command, you will be individually prompted for the source drive and then the destination drive or file name.

To re-create the lower density disk from a disk image file on the higher density disk of our imaginary drive, the command would be similar:

```
    DSKCOPY4 J:(filename) C:{RETURN}
```

Which creates a new low density disk in the C: drive from the disk image file that was stored on a high density disk in the same drive, but with imaginary drive letter J:.

Simulating Disk Formats:

Earlier we made mention of being able to create simulated 8" disks on 5.25" disks because they shared the same disk parameters.

Well, the program can actually be used to simulate ANY disk of a smaller size on ANY disk of a larger size. Can you think of any reason that we may wish to do this?

Well, suppose your friend does not use 8" disks, but you have a program on an 8" system disk that you would love for him to try. You could create a copy of the 8" disk for him on an 80 track 5-1/4" or even simulate the 8" disk on a 3-1/2" disk.

Or, suppose you use a GEMINI PC-emulator in your Z-100, which has internal 80 track 5-1/4" drives and an external 5-1/4" 360K drive configured as drive F:. The native mode recognizes all drives, but the GEMINI system can only use the 80 track drives as set by the motherboard DIP switch.

You have just received a later version of PC-DOS from e-bay, but they are on 360K disks and for some reason will not boot on the 80 track drives.

If the inability to boot was possibly the result of a disk read error, there is a possibility that using DSKCOPY4 with the /I switch (to set infinite retries) might permit you to make a new 360K system disk to try, or perhaps better, make a Disk Image File that could then be used to create a bootable 80 track disk.

We already discussed the procedures to make a new 40 track disk above. Let's try making an 80 track disk.

Note: As I do not have a computer configured for this example, the following steps have not been tried.

First, create a Disk Image File following the procedures provided earlier, but include the /I switch to set infinite retries:

DSKCOPY4 F: E:\(filename) /I{RETURN}

Then we create the desired disk from the disk image file.

If Drive A: was our 80 track 5-1/4" drive, to create a copy of the disk from our disk image file, use a command similar to:

DSKCOPY4 E:\(filename) A:{RETURN}

The computer will respond with:

Please insert source/destination disk(s)
and hit any key to continue....

Since we are using much different media in this example, the computer will display:

Destination device has at least twice as many tracks as source media. If you are attempting to create 40 track media in an 80 track device by double stepping, there is no guarantee of success if you are planning on reading it in a 40 track device.
You may also use only the first 40 tracks without double stepping.
Use ^C to Quit. Double Step media?(Y or N)>

Now, depending upon what you want to use the media you create in, you could either place the data on the first 40 tracks or double step - your choice. If you double step, you may also be able to read the disk in the 40 track drive, so I suggest using that.

The computer would continue with:

40 Tracks, Double Sided, 250K Transfer, 300rpm
Reading Source
E:\Image\(filename)
Writing Destination
Track XX Side X (action)
Copy completed successfully.

Note: The (action) above will alternate between displaying 'Formatting', 'Writing", and 'Verifying'.

Note: We would end up with a double stepped copy of our 40 track original disk. DSKCOPY4 will erase the data on the intermediate tracks in an effort to improve the disk's readability in a 40 track drive and reduce cross track interference in an 80 track drive.

Using FM/MFM Disk Encoding:

In "*Z-100 LifeLine*", issue #99, we briefly described the differences between FM and MFM encoding. Very early disk drives used FM encoding, but they were quickly replaced with the greater capacity MFM encoded formats. With the new **ZFMT207.COM** described in that issue, a disk could still be created using FM encoding.

This may be an advantage, being that few, if any, other computers will still recognize that format, making the disk very secure. The dis-advantage is that the disk capacity is about 50% of the MFM format.

Using an Imaginary Drive letter to define one of the disks, a disk image could be created of the FM encoded disk on an MFM encoded disk. The FM disk could then be re-created any time it was needed.

Using Expanded Disk File Labels:

As we could end up with images of dozens of disks on a hard drive, we wanted a better way to describe a disk image than a very limiting 8 character filename. So John has incorporated a well-thought out three-line heading for all Image Files that already contains the DISKCOPY version that created the file; the source disk's number of tracks, whether one or two sided, the transfer rate and rpm; and the Expanded Disk File Label containing from 0 to 79 characters.

A typical heading would be:

```
DiskCopy Version 4.06 Disk Image File
40 Tracks, Double Sided, 250K Transfer, 300rpm
(Expanded Disk File Label here)
```

The Expanded Disk File Label permits you the ability to personalize even more information to meet your needs. A suggested description might include the source disk's name or information about the formatting.

Examples are:

```
    5.25" CP/M-85 Boot Disk
    FM 5" DOS Steve Data
    5" FM/MFM CP/M Galahad Files
```

You may also want to settle on a standard extension for Disk Image Files, such as .IMG so that you can easily search for the Image Files. Other simple extensions might be: .IM3, .IM5, and .IM8, to differentiate between disk sizes more readily.

This heading information can be displayed by a couple of methods. The first three lines of Image Files are all ASCII text, with a simulated end-of-file marker at the end of the Expanded Disk File Label. This allows the TYPE command to show the first 3 lines of an Image File. For example, if you use TYPE on one of the Image Files created above:

TYPE B:TEST1.IMG{RETURN}

The computer will display something similar to:

DiskCopy Version 4.06 Disk Image File
40 Tracks, Double Sided, 250K Transfer, 300rpm
(5" FM/MFM CP/M Galahad File volume 3)

If there were several Image files on a disk, another method of display would be to use the COPY command, either to CON for just displaying the files, or to PRN to print out a listing. For example:

COPY B:*.IMG CON{RETURN}

will display the first three lines of each Image File on the screen. Here, an identifying extension comes in very handy.

Using SETIMAG:

We saw earlier how we can use DRIVECFG to map an imaginary drive to a physical drive. However, it is somewhat cumbersome to assign several imaginary drives and keep track of their assignments. Further, it is not very convenient to use DRIVECFG to temporarily change the mapping assignments.

Would it not be better to have one, or maybe two, imaginary drive letter(s) that can be reassigned to any physical drive as the situation warrants?

Well, John Beyers was way ahead of you. He has written a small utility that permits easily changing the imaginary drive's physical drive assignment. Once an imaginary drive letter has been initially mapped by using DRIVECFG, **SETIMAG.COM** can be run at any time to change the imaginary drive's physical drive assignment.

The initial setup is easy. From in the root directory, enter the command:

DRIVECFG{RETURN}

The DRIVECFG screen will list all the present drives, up to 26 drive letters, in the computer. Highlight the **"Next Available Drive to Assign"** and press {RETURN}. A window will open that lists the various drive types. Select **"(F0) Imaginary Drive"** and press {RETURN} again.

The computer will respond with:

Select Physical Drive Letter to map this Imaginary Device (ESC to cancel)...

Enter the drive letter of the Physical Drive that will be used most often. I assigned Imaginary Drive J: to Physical Drive A:, my 360K drive. Now press {RETURN} again.

You are returned to the drive listing and will see that an Imaginary Drive is now listed on the screen. Press {ESCape} and save the changes. Press {ESCape} again to quit. Finally, reboot the computer for the changes to take affect.

Now that an initial imaginary drive letter has been assigned, you can use John Beyers' utility, **SETIMAG.COM**, to change the imaginary drive's physical drive assignment as needed.

As with the other DOS utilities, if you enter the command:

SETIMAG [?] [or /?]{RETURN}

or if you should make an error using the command, a short help paragraph will be displayed:

This program will map imaginary drives to physical drives. Only command line input is permitted. Use DOS drive letters.
SETIMAG <imag drive> <real drive>

Therefore, to invoke SETIMAG.COM:

Usage: [a:\][pathname\]**SETIMAG** [?]
[<imag drive> <Real drive>]{RETURN}

Where anything in [brackets] is optional and:

a:\(path)	- The drive\pathname where the transient command file is located.
<imag drive>	- The Imaginary Drive Letter as assigned by DRIVECFG.
<real drive>	- The letter of any Physical Drive in the system.

Say we have used DRIVECFG to assign an Imaginary Drive Letter J: to Physical Drive A:, but we need to use our 3.5" drive, drive B:, for a particular operation. Using:

SETIMAG J: B:{RETURN}

will reassign our Imaginary Drive J: to Physical Drive B: until we run SETIMAG again to change it, or until we reBOOT, when IO.SYS will return to the drive arrangement as set by DRIVECFG.

Reference Information:

Definitions:

READ ADDRESS Command - Upon receipt of the READ ADDRESS command, the head is loaded and the Busy status bit set. The next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the data register. These six bytes are:

1) Track Address
2) Side Number
3) Sector Address
4) Sector Length
5) CRC 1
6) CRC 2

READ SECTOR Command - Upon receipt of the READ SECTOR command, the head is loaded, the Busy status bit set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct Cyclic Redundancy Check (CRC), the data field is read to the computer.

WRITE SECTOR Command - Upon receipt of the WRITE SECTOR command, the head is loaded, the Busy status bit set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, the data is written to the data field on the disk.

READ TRACK Command - Upon receipt of the READ TRACK command, the head is loaded and the Busy status bit set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register.

WRITE TRACK Command - Upon receipt of the WRITE TRACK command, the head is loaded and the Busy status bit set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse.

READING the SOURCE disk:

The READ ADDRESS command is used on each track /side to determine FM or MFM format and the track number, sector number, and size for each sector on the track.

If the /X switch is selected, the READ TRACK command is used to read the track format so it can be determined if there are non-standard sector sizes. A message is displayed to the screen if any are found. We might also discover sectors that the READ ADDRESS command missed.

Finally the READ SECTOR command is used to read the sector data and determine the record type (deleted or normal). See switch explanations for more information.

Read/Verify Error bits are defined as:

0) Busy	Command is under execution
1) DRQ	Data Request is pending
2) Lost Data	Computer did not respond to DRQ in one byte time
3) CRC	CRC error encountered in DATA, or ID field if RNF
4) RNF	Record Not Found, no ID found on track
5) Record Type	1 = Deleted Data Mark 0 = Regular Data Mark
6) Undefined	
7) Not Ready	Drive is not ready

WRITING the DESTINATION disk:

Initially, a RPM test is performed on the drive to determine how many bytes can be written to a track. This is used to construct each track image from the source sector information. An Index Address Mark is included if it will fit.

If the user does not select the /Y switch, the WRITE TRACK (format) command is first used on each track/side. If a sector's amount of data can be used with the WRITE TRACK command (no 0F5h-0FEh bytes), the actual data is written, otherwise the sector is filled with 0E5h's. Then the WRITE SECTOR command is used to put the sector data on the disk.

If actual data was written by the WRITE TRACK command, the sector write is skipped, but a verify existence is performed. If the actual data for all sectors on the track were written by the WRITE TRACK command, the write routine is completely skipped.

Finally, if the /V switch is not selected, the VERIFY SECTOR command is used to ensure the data was written correctly (a CRC is made).

The assumption is made that good media can be found, so all write errors abort DSKCOPY4. It is up to the user to assure correct media type is in the destination drive.

John has been unsuccessful at designing a test to reliably detect incorrect media type. In these cases, writing to the destination will eventually fail, but usually some data looks like it was put on the disk correctly.

Write Error bits are defined as:

0) Busy	Command is under execution
1) DRQ	Data Request is pending
2) Lost Data	Computer did not respond to DRQ in one byte time
3) CRC	CRC error encountered in DATA, or ID field if RNF
4) RNF	Record Not Found, no ID found on track
5) Write Fault	Write Fault error
6) Wrt Protect	Disk is hardware Write Protected
7) Not Ready	Drive is not ready

Additional Notes:

* Pressing a {Control}-{C}, (^C), at any time should always abort the program. Be mindful, however, that a partially created disk is of little use, and DOS will not be able to tell the last half is missing.

* Using a single disk drive is permitted, but you may not use the same MS-DOS drive letter. You must set up an imaginary drive letter with DRIVECFG. You can use SETIMAG to map this imaginary drive to any MS-DOS drive letter. The BIOS will prompt you when it is time to change media.

* When creating 40 track media in an 80 track device, 0's are now written in the skipped tracks created by double stepping. This seems to allow a high degree of success when reading the disk in a 40 track device.

* Many READ TRACK problems were encountered during the creation of DSKCOPY4. First and foremost is the inability to retrieve correct DATA and ID information in MFM mode. The hardware documentation assures that this information should be correct but the LifeLine Staff were unable to get it to work on multiple machines. The only ID problems seemed to be on track 41, but there could be more. The program now special cases retrieval of ID information on track 41 in MFM mode and the READ SECTOR command is now used to retrieve the DATA.

* If you want to make multiple copies of a disk, try creating a disk file on the hard drive. It does not take as long to read it. You could create a batch file that prompts for a new disk and automatically runs DSKCOPY4 again.

* A Disk Compare function was not added to DSKCOPY4 because both FC.EXE and COMP.COM will now compare multiple files on disks to see which files are different. You can also use DSKCOPY4 to make a disk image file for both disks and run COMP to see if they are exactly the same.

* Please remember that all Z-100 Operating Systems before Z-DOS v4 have no knowledge of 500K transfer rates on the 34-pin connector using the 5" FASTSTEP signal. They rely on the Motherboard DIP switch to inform them whether the computer is using 40 track or 80 track drives on the 34-pin connector. **GEMINI** and **EasyPC**, our PC-Emulators, which are using PC-DOS and their own ROM chips have the same limitation.

Disk Image File Format:

```
IDSTRING - "DiskCopy Version 4.06 Disk Image File"
"77 Tracks, Single Sided, 500K Transfer, 360rpm"
79 byte User Label to identify specific disk being
copied to this file.
Byte - Media Type/Transfer Rate(0=360rpm 250K,
       1=360rpm 500K,2=300rpm 250K,3=300rpm 500K)
Byte - # of Tracks (1 to 99)
Byte - # of Sides (1 or 2)
*** Repeated for each Track ***
Word - Total bytes this track
*** Repeated for each Side ***
Byte - FM (0) or MFM (1)
Byte - Sectors per side (0 to 68), 0=UNFORMATTED
*** Repeated for each Sector ***
Byte - Track number (0 - F4h) F5h-FFh cannot be
       written in ID header
Byte - Sector number (0 to 0F4h)
Byte - ID Sector size (0=128,1=256,2=512,3=1024)
Byte - FLAG
        00000001 Data ID (0=0FBh good, 1=0F8h deleted)
        00000010 Non-Standard Sec size
        00000100 Duplicate Sec # in track
        10000000 Sector DATA can be written by
                 write track
Word - REAL Sector size (can be less than or equal
       to ID header definition)
*** Sector Data for this side ***
```

Track Image Structure:

```
Byte - Track #
Byte - Sector #
Byte - Sector size (0=128,1=256,2=512,3=1024)
Byte - Read Address returned Status
Word - Real Sector size
Byte - Various flags defined in disk structure
         00000001 Data ID (0=0FBh good,
                  1=0F8h deleted)
         00000010 Non-Standard Sec size
         00000100 Duplicate Sec # in track
         00001000 Sector found by Read Address
         00010000 Sector found by Read Track
         10000000 Sector DATA can be written
                  by write track
Word - Ptr to ID in Track Image
Word - Ptr to DATA in Track Image
Byte - Read Sector returned Status
```

Note: The Track Image Structure is internal to DSKCOPY4 only, but is included here for informational purposes only.

Closing:

John Beyers is commended for creating such an important and useful utility. I believe that as magnetic floppy disk media continues to become more difficult to find and maintain, this utility will become a critical element in the future usefulness of the Z-100.

As I continue to work with DSKCOPY4, I'll try to report on additional examples that show the use of the switches.

On behalf of all of us... Thank You John.

If you have any questions or comments, please email me at:
z100lifeline@swvagts.com

Cheers,

Steven W. Vagts

[illegible]